63-2-3

MEMORANDUM
RM-3416-PR
JANUARY 1963

A PROGRAMMING SYSTEM FOR GENERAL NEURAL NETS

J. W. Smith

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-3416-PR
JANUARY 1963

A PROGRAMMING SYSTEM FOR GENERAL NEURAL NETS

J. W. Smith

This research is sponsored by the United States Air Force under Project RAND — Contract No. AF 49(638)-700 — monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Technology, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force. Permission to quote from or reproduce portions of this Memorandum must be obtained from The RAND Corporation.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

PREFACE

This Memorandum describes the computer programming system used in the study of general neural nets, presented in RM-3406-PR, A General Neural Net, E. W. Paxson and J. W. Smith, The RAND Corporation, November 1962.

The present Memorandum is concerned with the details of the electronic digital simulation of such a net, and serves as an adjunct to the earlier Memorandum. The study is part of the Project RAND research program in physiological communications systems.

SUMMARY

A computer programming system for investigating general neural net* activity is described. The system, called NET1, allows investigators to construct nets, maintain files of nets, and compose--and execute--pre-planned programs of experimentation on nets. Such experimental programs may effect repeated simulations of a single net or collection of nets with inter-simulation variation of net components, parameters, and stimuli; and may interrogate, test, and vary such elements within an individual simulation.

NET1 operates within the RAND version of the SHARE Operating System (SOS) for the IBM 7090.** Net capacity and ratio of computing time to physiological time are discussed for that machine.

---

*E. W. Paxson and J. W. Smith, A General Neural Net, The RAND Corporation, RM-3406.

**G. E. Bryan, Ed., The RAND-SHARE Operating System for the IBM 7090 Computer, The RAND Corporation, RM-3327.

CONTENTS

I. INTRODUCTION

A neural net is a set of nodes and their interconnections which purportedly have some of the observed properties of neurons and their axonal connections. A general neural net is a computer program which simulates the behavior of neural nets by computing the events that occur at neuronal synapses, within neurons, and along the connecting links.[1]

NET1 is a programming system for general neural net experimentation. It is operative on the IBM 7090, and requires the RAND version of the SHARE Operating System for that machine.

NET1 was designed to function as a technical assistant for neural net investigators. Such an assistant should be able to: construct nets from structural specifications, maintain files of nets, modify existing nets, and activate a specified net or collection of nets under prescribed conditions as directed by a set of instructions--testing, modifying, and recording behavior and conditions as directed. To these ends, NET1 contains the following elements:

1) A descriptive language for specifying the structure of nets in terms of neurons, connections, and stimuli.

---

[1] E. W. Paxson and J. W. Smith, A General Neural Net, The RAND Corporation, RM-3406.

2) An algorithmic language for composing experimental programs. Such programs may construct nets, modify nets, effect repeated experimental runs on nets with inter-run variation of net structure and environment, and may interrogate and vary structure, environment, and behavior within individual runs.

3) A collection of computer routines for performing the operations implied in 1) and 2) above.

The main concern of this Memorandum is the description of the languages available for the presentation of nets and experimental programs to NET1. In addition, the main neural net program, its capacity (in terms of neurons and connections), and its computing times will be discussed.

Terms introduced in context or defined in RM-3406 will be underlined the first time they occur. Abbreviations will be introduced in parentheses immediately following the first occurrence of a term.

II. NET SIMULATION

That computer routine which actually simulates the behavior of a net is called the simulator. Two were originally constructed and compared. One operated as a sequential simulator--one that computes the state of a net by systematically updating the entire net, neuron-by-neuron, at fixed intervals of time. The other functioned as an event simulator--one that computes the state of a net in a nonsystematic manner, updating only active neurons and connections (those that are either receiving, processing, or transmitting pulses). Figure 1 is a suitably linearized and normalized graph of computing time versus net activity for the two simulators; k% activity meaning that k% of all neurons and k% of all connections are active.

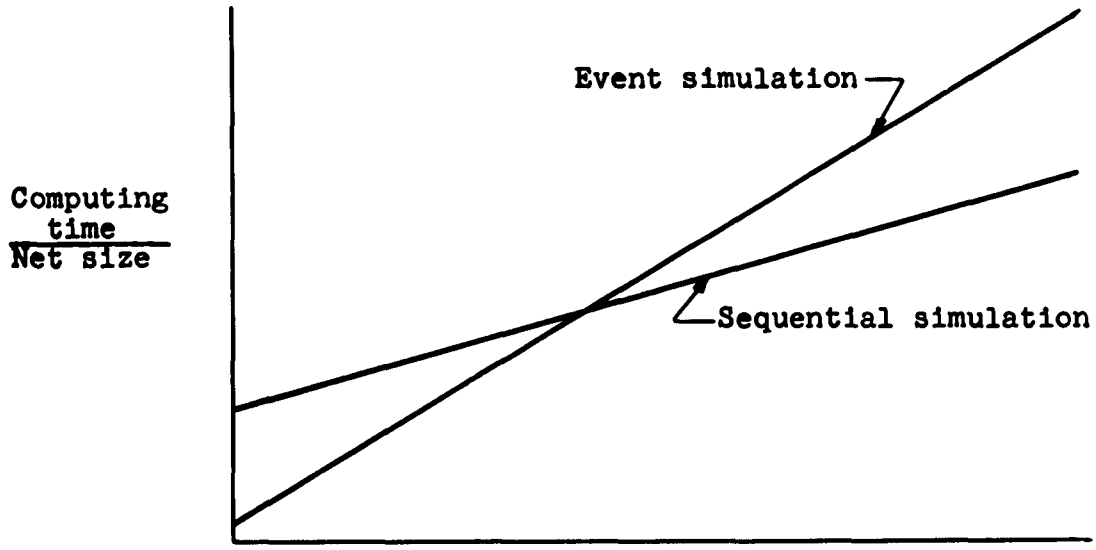

Fig. 1

A cross-over point, which is characteristic of such comparisons, occurs at an activity level of approximately 40%. It was felt that this level was too low to warrant the use of an event simulator for neural nets, which are generally characterized by rather high activity levels. Hence the simulation is a sequential one, from which it follows that:

$$\Delta T = k\tau,\ k \geq 1,\ \tau = \tfrac{1}{2} \text{ millisecond.} \qquad (2.1)$$

Here, ΔT is the updating interval, τ is the basic unit of physiological time, and k is the grain of the simulation, $k=1$ giving the finest grain.

Estimates of ratios of computing time to physiological time for varying net sizes, and estimates of maximal net sizes are presented in Appendix A.

III. NET DESCRIPTIONS

A net is described by listing its component neurons, connections, and stimuli. The list may contain display pseudo-components that provide, during the course of a simulation, stylized recordings of net behavior.[2] The components are, in effect, drawn from an alphabet of functional types, differentiation within a type being effected by parameterization. Tables 1, 2, and 3 (pp. 11, 13, 15, respectively) list available types, their functions as net components, and their parameters. All will be discussed, in appropriate detail, in succeeding sections.

NET1 expects the components of a net to be presented on standard IBM cards, one per card, in a format which is invariant over component types. The format, described in Fig. 2, is essentially functional notation; that is, a function name, followed by its arguments separated by commas.

card columns	8 - 15	16 - 72
contain	functional type	parameters - separated by commas - in the order specified by Tables 1, 2, and 3

Fig. 2

[2] See Sec. IV.

A collection of such cards, suitably delimited, constitutes a net description; the components may be presented in any order. A net description may be improperly delimited, may describe an oversize net, or may contain errors of omission, commission, and transcription; a net so described is ill-defined. NET1 will comment on such errors, taking appropriate action to insure that it accepts only well-defined nets.

Note that all components have one parameter type in common--an identifier or name. Indeed, in order that the connectivity of a net be unambiguously defined, it is necessary that its components be assigned unique names. Accordingly, the following naming convention--which must be honored--has been adopted:

1) Neuron names are drawn from an alphabet of alphanumeric couplets.
2) Connection names are triples of alphanumeric couplets of the form:

name of originating neuron
name of destination neuron
local identification.

For example,

FL LN S1 21 22

are permissible neuron names; whence,

FL2101 LN2201 S1LN01 222101

are proper connection names. The connectivity is pictured in Fig. 3.

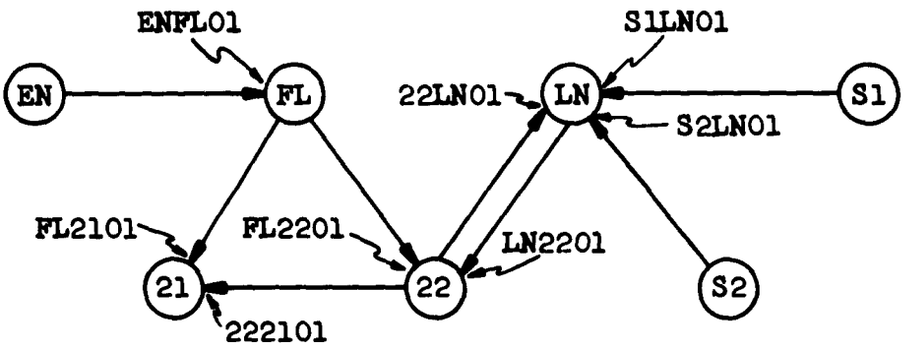

Fig. 3

Figure 4 is an example of a net description, describing the net presented diagramatically in Fig. 3. Note the delimiters **BEGIN** and **END**; these should have appended to them, as a parameter, the net name, which is necessary for future filing and simulation. Comment cards, indicated by the asterisk in column 1, have been liberally used for explanatory purposes. **SAMPLE**, the net described in Fig. 4, is well-defined.

Refer again to Table 1. Most of the functional types and parameters there listed are self-explanatory in the context of RM-3406. A discussion of the unfamiliar types and of parameter conventions follows.

Delay elements are single-input, multiple-output components whose sole function is to delay the transmission of pulses.

NET DESCRIPTION FOR NET OF FIGURE 3

```
      BEGIN   SAMPLE
      CYCLIC  EN,0,100,1     ZERO DELAY,
      CYCLIC  S1,0,100,1      100PPS RATE,
      CYCLIC  S2,0,100,1       NEVER FAIL.
*
* THBASE=11MV., THFUNCTION=TYPE3, RHO=2
*          FOR NEURON FL.
      NEURON  FL,11,TYPE3,2,*
*
* THBASE=10MV., THFUNCTION=TYPE1, RHO=2
*          FOR ALL OTHER NEURONS
      NEURON  21,10,TYPE1,2,*
      NEURON  22,10,TYPE1,2,*
      NEURON  LN,10,TYPE1,2,*
*
* P1 DISPLAYS FIRING HISTORIES OF 'MARKED'
* NEURONS,EXAMINING THESE NEURONS EVERY
* TIME PERIOD (RATE=2000PPS).
      PRINTC  P1,0,2000
*
* ALL END-POINTS FIRE WITHOUT FAIL,I.L.
* RELIABILITY=1 FOR ALL CONNECTIONS
      FACIL   ENFL01,1,1,15      DELAY=1,AMP=15MV
      PULSE   FL2101,1,8,15            =8     =15
      PULSE   FL2201,1,4,14.5          =4     =14.5
      PULSE   222101,1,4,-15           =4     =-15
      PULSE   22LN01,1,1,4             =1     =4
      PULSE   LN2201,1,1,14.5          =1     =14.5
      PULSE   S2LN01,1,8,20            =8     =20
      REINF   R1,22LN01,S2LN01
*
* S1LN01 REQUIRES COINCIDENCES WITH BOTH 22LN01
* AND S2LN01. THE FORGETTING FACTOR IS TO BE
* VARIED BETWEEN RUNS.
      LEARN   S1LN01,1,16,20,R1,1.1,2
      END     SAMPLE
```

FIGURE 4

Reinforcing sets are collections of from one to six connections that act as a reinforcing consortium; learners that depend on such sets require a specified number of simultaneous coincidences to trigger an effective coincidence--one that actually increments the learning function. Learners depending on reinforcing sets, and only such learners, must specify this coincidence count as a parameter.

Learners may specify individual forgetting rates (λ), a 'normal' forgetting rate for learners being specified as a net parameter.[3] Omission of λ from the parameter list indicates that the normal rate is to hold; when specified, λ will modify individual forgetting rates according to the following schema.

$$\lambda \begin{Bmatrix} > 1 \\ = 1 \\ < 1 \\ = 0 \end{Bmatrix} .\supset. \begin{Bmatrix} \text{higher forgetting rate (slow learning)} \\ \text{normal forgetting rate} \\ \text{lower forgetting rate (fast learning)} \\ \text{no forgetting (incremental learning)} \end{Bmatrix}$$

It should be noted that any connection may be a reinforcer. Indeed, learners may themselves be reinforcers, with no limit to the length of such reinforcing chains.

All delays (δ) must be given as an integral number of basic time units ($\tau = \frac{1}{2}$ ms). That is, if the

---

[3] See Sec. VI.

true delay were t seconds, then δ would be given by:

$$t = \delta.\tau \qquad (3.1)$$

with the further restriction that

$$1 \leq \delta \leq 35 \ . \qquad (3.2)$$

This constraint may be circumvented by cascading delay elements.

The absolute refractory period (ρ) must obey:

$$t = \rho.\tau \qquad (3.3)$$

where t is the refractory period in seconds.

The threshold resting value (th_0), pulse amplitude (A), and stabilizing level (L) are given in millivolts, and may take on any reasonable value.

Note that inhibitory pulses are indicated by negative A's.

The threshold function is given by name; three are currently available. They are--by name--TYPE1, TYPE2, and TYPE3, corresponding to those described in RM-3406. Other threshold functions may be constructed as described in Appendix B.

Neurons whose firing histories are to be displayed during the course of a simulation must have some non-zero mark appended to their parameter lists--as an additional parameter. An asterisk (*) has been used in the example of Fig. 4.

Table 1

FUNCTIONAL TYPES AND PARAMETERS

Type	Function[a]	Parameters
NEURON	Neuron	Name, threshold resting value, threshold function, absolute refractory period
DELAY	Delay element	Name, delay
SQUARE	Square wave	Name, probability of end-point firing, transmission-line delay, pulse amplitude
PULSE	Standard pulse	ditto
FACIL	Facilitory pulse	ditto
DEFAC	Defacilitory pulse	ditto
STABIL	Stabilizing pulse	Name, probability of end-point firing, transmission-line delay, pulse amplitude, stabilizing level
NULL	Input to delay element	Name, probability of end-point firing, transmission-line delay
LEARN	Learning connections	Name, probability of end-point firing, transmission-line delay, pulse amplitude, name of reinforcing connection or set of connections, forgetting factor, number of simultaneous coincidences with members of reinforcing set required to trigger an effective coincidence
REINF	Set of reinforcing connections	Name, names of all connections in set
BASE	Square wave which persists until parent neuron fires	Name, probability of end-point firing, transmission-line delay, pulse amplitude

[a]See RM-3406.

IV. STIMULI

These functional types are listed in Table 2; they are treated as pseudo-neurons with neuronal naming conventions.

All times are given in seconds, all rates are given in pulses-per-second (pps), and all reliabilities, which specify the probability of transmission of impulses into the net from the particular stimulus, must lie in the closed interval [0,1] .

The general pattern of firing is: after an initial waiting period specified by the pre-firing time (t_{pf}), firing is initiated; firing then continues at the specified rate for a period of time given by the burst time (t_{on}), after which firing is delayed for the inter-burst time (t_{off}); the on-off cycle then repeats. Methods for controlling stimuli in a less stylized manner are described in Sec. VII.

INPUTS controls the firing by a specified function; the construction of such special-purpose functions is discussed in Appendix B.

BURSTS is clearly a special case of INPUTS, while CYCLIC is a specialization of BURSTS with $t_{off} = 0$ and t_{on} a very large number.

RANDOM is a further specialization of CYCLIC with firing rate set at 2000 pps (firing at intervals of τ).

Table 2

STIMULI TYPES AND PARAMETERS

Type	Function	Parameters
CYCLIC	Continuous firing at constant rate	Name, pre-firing period, firing rate, firing reliability
BURST	Bursts of continuous firing	Name, pre-firing period, firing rate, firing reliability, burst period, inter-burst period
RANDOM	'Random' firing	Name, firing reliability
INPUTS	Burst firing at rate determined by special function	Name, pre-firing period, name of function, firing reliability, burst period, inter-burst period, two or fewer parameters for the function

V. DISPLAY

The pseudo-components listed on Table 3 are used to display net status and firing histories at specified intervals. The display formats are described in Appendix C. The parameters of Table 3 are in one-to-one correspondence with those of Table 2, with 'display' replacing 'firing'.

PRINTA and PRINTB display a rather complete picture of the state of a net (see Appendix C), the former operating in a 'burst-wise' manner, the latter operating under the aegis of a special function.

PRINTC displays firing histories for all stimuli and all 'marked' neurons (see Sec. III and Appendix C).

PRINTD is not used for display purposes; rather, it is used to save--on magnetic tape--the complete status of a net for future editing or restart.

Table 3

DISPLAY TYPES AND PARAMETERS

Type	Function	Parameters
PRINTA	Display state of net in bursts	Name, pre-display period, display rate, burst period, inter-burst period
PRINTB	Display state of net via special function	Name, pre-display period, name of function, burst period, inter-burst period
PRINTC	Display firing histories	Name, pre-display period, rate at which neuron firings are examined
PRINTD	Save complete net for restart	Name, pre-operative period, number of times per second net is to be saved

VI. NET PARAMETERS

Net parameters are computational and 'physiological' factors which specify a common mode of behavior for all net components, as described in Table 4. The normal values there listed remain in effect unless changed by the net program (see Sec. VII).

Table 4

NET PARAMETERS

Name	Normal Value	Function
T1	0	The lower coincidence
T2	10	upper limits (as an integral number of btp's) for learning connections[a]
LAMBDA	.0005	The normal forgetting rate
MUO	5	Learning parameters[a]
MU1	15	
MUSTAR	1	
GRAIN	1	The grain of the computation
START	0	Time for initiation of processing
KAPPA1	5/3	Parameters for facilitory/ defacilitory connections[a]
KAPPA2	1/60	
KAPPA3	1/3	
KAPPA4	1/60	
SIGMA	SIGMA1[a]	Synaptic delay function
RZERO	1111111111	An initial random number which must be large and odd

[a]See RM-3406.

VII. NET PROGRAMS

The descriptive language for communicating net structures to NET1 has been described; there remains the algorithmic language for communicating programs of experimentation to be carried out by NET1. Such programs are presented as an ordered set of directives or instructions, each such instruction specifying a task to be performed and, when necessary, the entities on which the task is to be performed. That is, an instruction is composed of an operation and operands, or equivalently, a function and arguments. Many instructions are concerned primarily with operations on net components and net descriptions; others allow simple arithmetic operations to be performed, allow testing and interrogation, and allow the user to specify the order in which NET1 is to carry out the instructions. The various classes of instructions will be discussed below; but first, a brief uberblick.

Nets originally exist on cards as net descriptions; let us call a set of one or more such descriptions an input stream. Net descriptions may be called from the input stream (sequentially), may be saved on a description file (magnetic tape), and may be recalled from a description file; files may be labelled and saved from run to run. Once called, a net description may be renamed and modified. Modification may involve appending

new components to the net description, amending existing components (in toto), and changing specific component parameters and component state variables.[4] Net descriptions, components, component parameters, and state variables may all be displayed, interrogated, and tested. Net descriptions may be activated and run for specified periods of time. Activated nets cannot be saved on a description file, but may be saved on a record file for future reactivation and/or editing. All other operations on net descriptions may be applied to activated nets. In addition, all nodes--neurons, stimuli, and display--in activated nets may be turned off and on. Net parameters may be modified, tested, and displayed at any time.

A net program is a suitably delimited set of instructions chosen from the repertoire of Table 4 and/or the SCAT[5] repertoire. Instructions are presented to NET1 in standard form, with the additional proviso that columns one through six are reserved for the (unique) identification of any instructions referenced within the net program. Such identifiers will be referred to as such.

A net program, possibly followed by one or more net descriptions, is presented to NET1 as a set of cards.

---

[4] See Table 5.

[5] International Business Machines Corporation, SHARE SOS Reference Manual--SHARE Operating System for the IBM 709, IBM Applied Programming Publications, New York, 1960-61.

Table 5

COMPONENT PARAMETERS AND STATE VARIABLES

Name	Description
NAME	Component name
RELIAB	The 'reliability' of a connection--the probability that end-point firing will occur
AMP	Connection pulse-amplitude
COUNT	Coincidence-count for learners
DELAY	Connection transmission-line-delay
DELAY	Delay-element delay
FORGET	Forgetting-factor for learners
LEVEL	Stabilizing-level for STABIL connections
MU	Learning-factor for learners
REINF	Name of reinforcing knob or set--for learners
POT	Neuron membrane potential
RHO	Neuron refractory-period
THBASE	Neuron threshold-base-value
THTYPE	Neuron threshold-function
FIRE	Number of times neuron or stimulus has fired
FUNCT	Name of special purpose function
PARAM1 PARAM2	Parameters of special functions for stimuli and display
PROBAB	Stimulus reliability
RATE	Firing rate
TOFF	Inter-burst period
TON	Burst period
TPF	Time before next firing
TLEFT	Time before end of burst period

NET1 carries out the intent of the program by executing the instructions sequentially, beginning with the first instruction. Changes in sequencing are effected unconditionally by GOTO, DO, and LEAVE instructions, and conditionally by TEST instructions and by error conditions.[6] Some instructions must have, as an argument, an identifier specifying the instruction to be executed in the event of an abnormal condition. Use of the identifier EXIT for such arguments will cause the execution of the program to be curtailed by any error condition associated with the instruction. NET1 will comment on error conditions before executing the instruction supplied as an "error-exit"; the commentary should, in all cases, be sufficient to explain the error.

Figures 5, 6, and 7 (pp. 23-25) exhibit three distinct sets of instructions for performing the same experiment. Each experiment consists of three runs on the net, SAMPLE of Fig. 4, with variation of the forgetting factor for the learning connection. Each run lasts 3/10 sec., different stimuli phasing in at 1/10-sec. intervals. Firing counts for neuron 22 are displayed every ten seconds, and the learning factor for the learning connection is displayed at the end of the run.

---

[6] See Table 6, p. 26-27.

The examples show that modification of net elements--components, parameters, and state variables--is accomplished by referencing names, numbers, and elements incorporated within the net program. In particular, note that 'indexing' is allowed. That is, if I and J are identifiers and k is any integer, then

$$\left.\begin{matrix} I,k \\ I,-k \end{matrix}\right\} \text{ refers to the } k^{th} \text{ instruction } \begin{matrix} \text{after} \\ \text{before} \end{matrix} \text{ I}$$

and

I,J refers to the $(J)^{th}$ instruction before or after I,

where

(J) means 'the quantity identified by J.'

One caveat: it is essential that 'lists' of instructions to be referenced by indexing be coherent; that is, any such list must be a list of names and/or numbers, a list of elements, a list of GOTO instructions, or a list of DO instructions. Indexing over incoherent instruction lists will not be detected by NETl, and will result in unpredictable errors.

Program 1 of Fig. 5 illustrates the use of most of the instruction classes of Table 6. The first instruction states that a description file is to be made

available. In Program 1, the file is to be used to save the net SAMPLE between runs; that is, the file is used as a temporary storage medium. The net is first called from the input stream, then displayed by the PRINT instruction and is then saved on the file. Control then passes to TEST2, where the forgetting factor (originally 1.1) is set and displayed. The run is then carried out in one-tenth second intervals, after which the forgetting factor is decremented by 1/10 and is tested. If the forgetting factor is greater than or equal to 9/10, control passes to TEST1, where the net is recalled and then to TEST2 to begin the new run. The displays generated by the three runs are given in Appendix C.

Program 2 of Fig. 6 illustrates the use of subroutines, of indexing, and of the REFER instruction. Program 3 of Fig. 7 is a variant of Program 1. Note that both Program 2 and Program 3 save the description file.

```
*                    PROGRAM 1
        READY   FILE                    DESC. FILE TO BE USED
        CALL    SAMPLE,EXIT,CARDS       CALL NET FROM INPUT STREAM
        SAVE    SAMPLE                  SAVE NET FOR REUSE
        GOTO    TEST2
TEST1   CALL    SAMPLE,EXIT             CALL FOR NEXT RUN
*                               EXIT    PROGRAM ON ERROR CONDITIONS
TEST2   GET     TEST3                   FORGET(S1LN01)=NEW VALUE
        STORE   FORGET,S1LN01           (ORIGINALLY = 1.1)
        STOP    S1                      RUN FOR 1/10 SECOND WITH FL
        STOP    S2                            AS SOLE STIMULI
        PROBE   S1LN01                  PRINT FIRING HIST.
        ACTIVE  SAMPLE,TENTH,EXIT
        PRINT   TEST3
        PRINT   FIRE,22
        START   S1                      ACTIVATE S1
        CONT    TENTH,EXIT
        PRINT   FIRE,22
        START   S2                      ACTIVATE S2
        CONT    TENTH,EXIT
        PRINT   NET
        GET     TEST3                   DECREMENT LEARNING FACTOR
        MINUS   TENTH
        PUT     TEST3
*
* CONTINUE (RETURN TO TEST1) IF MORE RUNS, OTHERWISE EXIT.
        TEST    FINAL,TEST1,EXIT,EXIT
TENTH   NUMBER  .1
TEST3   NUMBER  1.1                     LEARNING FACTOR (TO BE DECREMENTED)
FINAL   NUMBER  .8
```

FIGURE 5

```
*                    PROGRAM 2
        READY   FILE
        CALL    SAMPLE,EXIT,CARDS
        SAVE    SMPL1                   FORGET(S1LNO1) = 1.1
        AMEND   FIRST
        SAVE    SMPL2                   FORGET(S1LNO1) = 1
        AMEND   SECOND
        SAVE    SMPL3                   FORGET(S1LNO1) = .9
        REFER   TESTA
        DO      RUN
        REFER   TESTB
        DO      RUN
        REFER   TESTC
        DO      RUN
        CLOSE   FILE,FILE2              SAVE DESC. FILE
        EXIT
*
*  RUN IS USED AS A SUB-ROUTINE WHICH OPERATES ON THE
*  NET LAST REFERRED TO BY A REFER INSTRUCTION.
        ENTER   RUN
        CALL    *,EXIT                  CALL NET BEING REFERRED TO
        PRINT   FORGET,S1LNO1
        STOP    S1
        STOP    S2
        PROBE   S1LNO1
        ACTIVE  *,TENTH,EXIT
        PRINT   FIRE,22
        START   S1
        CONT    TENTH,EXIT
        PRINT   FIRE,22
        START   S2
        CONT    TENTH,EXIT
        PRINT   FIRE,22
        PRINT   MU,S1LNO1
        LEAVE   RUN
FIRST   LEARN   S1LNO1,1,16,20,R1,1,2
SECOND  LEARN   S1LNO1,1,16,20,R1,.9,2
TESTA   NAME    SMPL1
TESTB   NAME    SMPL2
TESTC   NAME    SMPL3
TENTH   NUMBER  .1
```

FIGURE 6

```
*                    PROGRAM 3
         READY    FILE
         CALL     SAMPLE,EXIT,CARDS
         SAVE     SAMPLE
         ZERO     COUNT1
         GOTO     TEST3
TEST1    CALL     SAMPLE,EXIT
TEST3    GET      FG,COUNT1
         PUT      FORGET,S1LNO1
         PRINT    FORGET,S1LNO1
         STOP     S1
         STOP     S2
         PROBE    S1LNO1
         ACTIVE   SAMPLE,TENTH,EXIT
         PRINT    FIRE,22
         START    S1
         CONT     TENTH,EXIT
         PRINT    FIRE,22
         START    S2
         CONT     TENTH,EXIT
         PRINT    FIRE,22
         PRINT    MU,S1LNO1
         GET      COUNT1
         PLUS     ONE
         PUT      COUNT1
         TEST     THREE,TEST4,TEST1,TEST1
TEST4    CLOSE    FILE,FILE2
         EXIT
COUNT1   NUMBER
ONE      NUMBER   1
THREE    NUMBER   3
TENTH    NUMBER   .1
FG       NUMBER   1.1
         NUMBER   1
         NUMBER   .9
```

FIGURE 7

Table 6

NET1 INSTRUCTION REPERTOIRE

INST	ARGUMENTS	INTENT
CALL	net name, E	Call net from description file
CALL	net name, E, CARDS	Call net from input stream
SAVE	net name, E	Save net on description file
ACTIVE	net name, I, E	Activate net, run for (I) seconds
CØNT	I,E	Continue run for (I) seconds
APPEND AMEND	I' I'	Component identified by I' is amended, or is appended to net
PRØBE	C1, C2, ..., C6	Firing histories for C1, C2, etc., are to be displayed
GET	I'	(I') ⟶ (PR)
PUT	I'	(PR) ⟶ (I')
PLUS	I'	(PR) + (I') ⟶ (PR)
MINUS	I'	(PR) - (I') ⟶ (PR)
TIMES	I'	(PR) x (I') ⟶ (PR)
DIVIDE	I'	(PR) / (I') ⟶ (PR)
ZERØ	I'	0 ⟶ (I')
SETPØS	I'	|(PR)| ⟶ (PR)
SETNEG	I'	-|(PR)| ⟶ (PR)
FETCH	V,C	V [C] ⟶ (PR)
STØRE	V,C	(PR) ⟶ V [C]
TEST	I', I1, I2, I3	Next instruction is I1 if (PR) > (I') I2 if (PR) = (I') I3 if (PR) < (I')
TEST	I1, I2, I3	Next instruction is I1 if (PR) > 0 I2 if (PR) = 0 I3 if (PR) < 0
FETCH	π	(π) ⟶ (PR)
STØRE	π	(PR) ⟶ (π)
PRINT ⇕ PRINT	C I π V,C NØDES KNØBS PARAMS NET NØDE, node name KNØB, connection name	Self-explanatory, see Appendix C for formats

Table 6--Continued

INST	ARGUMENTS	INTENT
GØTØ	I'	Next instruction is I'
DØ	I	Execute sub-routine I; continue with next instruction upon completion of sub-routine
ENTER	sub-routine name	Defines beginning of sub-routine
LEAVE	sub-routine name	End (last instruction) of sub-routine
EXIT		End of net program
NUMBER	any number	E.g., {3, 1.2, 0, 14.3}
NAME	any name	E.g., {22, FL, FL2201}
CLØSE	FILE, file name	The description file is to be labeled and saved--no further operations may be performed on the description file by the current program
READY	FILE	A description file is to be 'opened'
READY	FILE, file name	The description file named (presumably saved from a previous run) is to be opened
REFER	I'	Future use of an asterisk (*) for a net name in CALL or ACTIVE will indicate that the name identified by I' is to be used as the net name
STATE	function name	Used to declare special functions
STØP	node name	Node is inactivated } primarily for stimulus and display
START	node name	Node is reactivated } primarily for stimulus and display
UNTIL	node name, I, E	Continue run until node named has fired (I) times

π	means	"any net parameter"
V	means	"any variable listed in Table 5"
C	means	"any component name"
E	means	"the identifier of the instruction to be performed in the event of an error"
(PR)	means	"the result of the previous arithmetic operation"--the "previous result"
I'	refers to any of	I ; I,K ; I,-K ; I,I
V[C]	means	"the variable, V, associated with component C"
(I)	means	"the quantity identified by I"

Appendix A

NETWORK CAPACITY AND PROCESSING TIMES

NETWORK CAPACITY

Of the 32,768 stores available in the 7090, approximately 5500 are usurped by the machine operating system and 5000 by NET1. Thus, about 22,200 are available for node/connection storage--at 12 stores per component--and for net programs. Distribution of available space among nodes and connections is not fixed a priori, but calculated anew for each run. Thus, no fixed maximum number of components can be stated. However, assuming six connections per node, a network of, say, 250 nodes and 1500 connections would leave ample room for net programs.

PROCESSING TIMES

The processing time--per network time period--is a function of the size of the net, of the types of connections, and of the states of cells (nodes) and knobs (connections). Cells may be in one of three states:

C_1 ~ Threshold reached

C_2 ~ Refractory

C_3 ~ Nonrefractory

with transition diagram:

$$C_1 \longrightarrow C_2 \longrightarrow C_3 \longrightarrow C_1 .$$

C_1 is a special case of C_3; the transition $C_1 \longrightarrow C_2$ is characterized by the processing time required to calculate synaptic delay and to place signals on the cell's efferent lines; this time can be neglected by amortizing it over the time the cell spends in state C_2 . The transition $C_2 \longrightarrow C_3$ involves a resetting of knob states; this time may be amortized over time in state C_3 . Knobs may be in one of twelve states, K_{mnp}, where m is the number of pulses active at the knob, n is zero or one as a signal is or is not arriving at knob, and p is one or two as the parent cell is or is not refractory. Processing times for states corresponding to $n = 1$ are equivalent to those for $p = 2$, and will be so aggregated; hence, only six states will be timed. The times below are given in terms of machine cycles--2.18μs.

a) Cells: 53 cycles if in state C_2
29 cycles " " C_1

b) Knobs: 22 cycles if parent cell in state C_2
16 cycles " " C_1

plus the following times (in machine cycles) dependent on connection type:

Type 10

m \ p	1	2
0	2	6
1	2	16
2	11	19

Types 20, 30

m \ p	1	2
0	0	6
1	0	28
2	0	36

Types 21, 22, 31, 32

m \ p	1	2
0	0	6
1	0	28
2	0	49

Type 40

m \ p	1	2
0	0	6
1	0	28
2	0	50

Learning

m \ p	1	2
0	96	102
1	96	124
1	132	132

Make two assumptions:

1) Cells fire every twenty time periods; hence ratio of refractory to nonrefractory time is approximately 6:1
2) Knobs have zero, one, two pulses active in the ratio 1:2:1

and obtain the following average times:

a) Cells ~ 50 machine cycles

b) Knobs ~ 21 machine cycles
Type 10 ~ 13 " "
Type 20 ~ 21 " "
Type 21 ~ 24 " "
Type 40 ~ 24 " "
Learning ~ 118 " "

Converting machine cycles to actual time and letting N_c, N_k, N_{10}, ... refer to total numbers of cells, knobs, Type 10 connections, ..., the processing time, T, per time period is:

$$T = (2.18)\{50N_c + 21N_k + 13N_{10} + \ldots\}\ \mu s. \quad (A.1)$$

The estimate is probably high, since assumptions 1) and 2) imply fairly high network activity.

A rough time-estimate for general nets may be obtained by assuming the following distribution of connection types:

60% Standard pulses
10% Learning knobs
10% Facilitory/Defacilitory knobs
10% Stabilizing knobs
10% Rectangular pulses

obtaining an "average" T given by

$$T_{avg.} \doteq (109)\ (N_c + N_k)\ \mu s\ . \quad (A.2)$$

For a network of 250 cells and 1500 knobs, T_{avg} = .193 sec., which is unconscionably high, since it means that the ratio of machine time to "physiological" time is 400:1 !

Appendix B

CONSTRUCTION OF SPECIAL FUNCTIONS

Subroutines for all functions must be written in SCAT,[7] with the following boundary conditions:

1) Threshold functions:
   a) first instruction is 'BEGIN 1,7'
   b) (IR2) on entry contains index to current node, so that THBASE (floating point) is obtained by addressing 'THBASE,2'
   c) (IR1) on entry contains time elapsed since last firing, as an integral number of basic-time-units ($\frac{1}{2}$ ms.)
   d) (ACC) on exit contains TH (floating point)
   e) exit via "RETURN' instruction.
2) Display and stimulus functions:
   a) first instruction is 'BEGIN 3,7'
   b) (IR2) on entry contains index to node
   c) PARAMi of subroutine is fetched via 'CLA* i,4' with (IR2) = (IR2) on entry

---

[7] See footnote 5, p. 18.

d) (ACC) on entry contains time--
in seconds--since beginning of firing
(floating point)

f) exit via return instruction.

3) Synaptic-delay functions:

a) first instruction is 'BEGIN 1,7'

b) (IR1) on entry contains time
elapsed since last time threshold
was exceeded, as an integral number
of basic-time-units

c) (ACC) on exit contains synaptic
delay as an integral number of basic-
time-units (fixed-point).

NET1 must be apprised of the existence of any special subroutines. This is effected by use of 'STATE' instructions, which--when used--must be the first instructions in the net program.

Appendix C

DISPLAY FORMATS

Three distinct categories of display format exist--display of value, display of component status, display of firing histories. Examples of each type are included in Figs. 8-12, which give displays resulting from a partial run of Program 1 of Fig. 5 on the network, SAMPLE, of Fig. 4.

A PRINT instruction, whose argument is the name of a state variable, a parameter, or a net program variable, displays on a single line:

time of display; name of variable; value of variable.

Figure 8 illustrates the format; the first line resulted from a PRINT TEST3 instruction, the last three from a PRINT FIRE,22 instruction executed at 1/10-second intervals.

Figures 9-11 are the result of a PRINT NET instruction, which is equivalent to the instruction triplet: PRINT PARAMS, PRINT NODES, PRINT KNOBS. Fig. 9 is self-explanatory. Each line of Figs. 10 and 11 is the result of a component display; for example, Fig. 10 is equivalent to the sequence of instructions: PRINT EN, PRINT S1, PRINT S2, etc. The general pattern for components is given in Table 7 below. Names of state variables (see Table 5) have been used. Asterisks (***) preceding a node

name indicate that the node has just fired. The delay line picture, Δ, is a 12-digit octal representation of a 36-bit binary number corresponding to a delay line 36 BTP's in length. Reading the number as $d_1\ d_2\ \ldots\ d_{36}$, a one (1) in position d_i indicates the presence of a signal that will reach the connection in i BTP's.

Figure 12 is a partial result of a display of firing histories (by a PRINTC display node). Each line gives a 100-BTP history of the component named--beginning at the time indicated at the head of the display. Dots (.) indicate no fire; dollar signs ($) indicate fire. Fig. 12 indicates that node EN has fired at .1995 seconds and continues to fire every 20 BTP's; i.e., at a rate of 100 pulses per second.

```
                NETWORK      SAMPLE
TIME =      .100       TEST3(     ) =      .90000
TIME =      .100        FIRE(     22) =          0
TIME =      .200        FIRE(     22) =          0
TIME =      .300        FIRE(     22) =    5.00000
```

Fig. 8.

```
NETWORK      SAMPLE
  TIME =      .30050
 START =           0
 GRAIN =     1.00000
    T1 =           0
    T2 =    10.00000
    K1 =     1.66667
    K2 =      .01667
    K3 =      .33333
    K4 =      .01667
LAMBDA =      .00050
   MU0 =     5.00000
   MU1 =    15.00000
MUSTAR =     1.00000
 SIGMA = SIGMA1
```

Fig. 9.

NETWORK SAMPLE

TIME = .30050

***	BURST	EN	.01000	100.00000	1.00000	0	31.00000	0	0	
	BURST	S1	.00050	100.00000	1.00000	0	20.00000	0	0	
	BURST	S2	.00050	100.00000	1.00000	0	10.00000	0	0	
	NEURON	FL	0	15.00000	0	0	15.00000	11.00000	2.00000	TYPE3
	NEURON	21	0	42.00000	-3.22500	-1.98000	10.00000	10.00000	2.00000	TYPE1
	NEURON	22	0	7.00000	0	0	5.00000	10.00000	2.00000	TYPE1
	NEURON	LN	0	27.00000	7.99600	9.24000	5.00000	10.00000	2.00000	TYPE1
	PRINTC	P1	92233719+11	2000.00000	0	0	0	0	0	

Fig. 10.

NETWORK SAMPLE

TIME = .30050

	FACIL	ENFL01	0	0	15.0000	0	15.0000	1.0000	0	1.0000	000000000000
	PULSE	FL2101	9	0	15.0000	0	15.0000	1.0000	0	8.0000	000000000000
	PULSE	FL2201	0	0	14.5000	0	14.5000	1.0000	0	4.0000	000000000000
	PULSE	2221G1	5	0	-15.0000	0	-15.0000	1.0000	0	4.0000	000000000000
	PULSE	22LN01	8	0	4.0000	0	4.0000	1.0000	0	1.0000	000000000000
	PULSE	LN2201	0	0	14.5000	0	14.5000	1.0000	0	1.0000	000000000000
	PULSE	S2LN01	12	0	20.0000	0	20.0000	1.0000	0	8.0000	000000000000
	REINF	R1	22LN01	S2LN01							
	LEARN	S1LN01	0	0	4.6796	-2.0000	20.0000	1.0000	.9000	16.0000	000000000000 R1

Fig. 11.

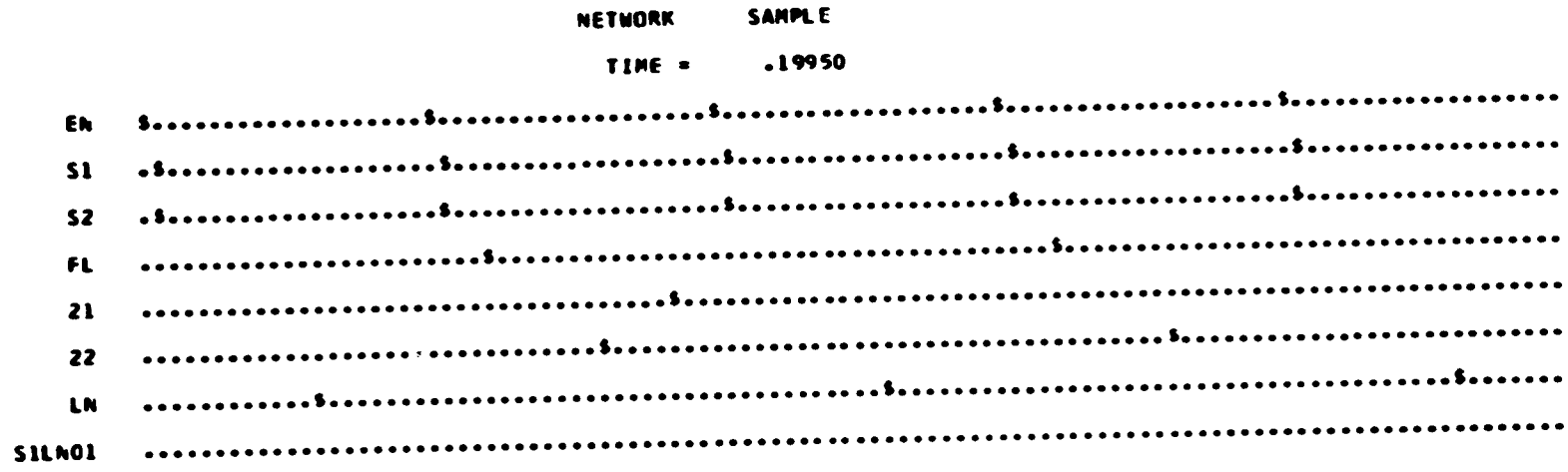

Fig. 12.

NEURON	COMPONENT NAME	τ_1	τ_2	POT	PREVIOUS MEMBRANE POTENTIAL	FIRE	THBASE	REO	THTYPE		
BURST	COMPONENT NAME	TPP	RATE	PROBAB	——	FIRE	TON	TOFF	FUNCT		
CONNECTION TYPE	COMPONENT NAME	δ_1	δ_2	a_1	a_2	AMP	RELIAB	LEVEL	DELAY	Δ	
LEARN	COMPONENT NAME	δ_1	δ_2	MU	COUNT	AMP	RELIAB	FORGET	DELAY	Δ	REINF

$\tau_1 = \tau_2 = 0$ ⊃· node has never fired.

$\tau_1 = 0$ ⊃· τ_2 = time since last firing (BTP's).

$\tau_1 \neq 0$, $\tau_2 \neq 0$ ⊃· node in synaptic delay period, τ_1 = time to end of period (BTP's).

$\tau_1 \neq 0$, $\tau_2 = 0$ ⊃· node in absolute refractory period, τ_1 = time to end of period (BTP's).

$\delta_1 = \delta_2 = 0$ ⊃· no pulses active at connection.

δ_1 = time since initiation of most recent pulse (BTP's).

δ_2 = time since initiation of previous pulse (BTP's).

a_1 = amplitude of most recent pulse.

a_2 = amplitude of previous pulse.

Δ = delay line picture (see text).

Table 7

COMPONENT DISPLAY FORMATS